Estimating the Bayes Point Using Linear Knapsack Problems

Brian Potetz POTETZ@ITTC.KU.EDU
University of Kansas, Lawrence, KS 66045 USA

Abstract

A Bayes Point machine is a binary classifier that approximates the Bayes-optimal classifier by estimating the mean of the posterior distribution of classifier parameters. Past Bayes Point machines have overcome the intractability of this goal by using message passing techniques that approximate the posterior of the classifier parameters as a Gaussian distribution. In this paper, we investigate alternative message passing approaches that do not rely on Gaussian approximation. To make this possible, we introduce a new computational shortcut based on linear multiple-choice knapsack problems that reduces the complexity of approximating Bayes Point belief propagation messages from exponential to linear in the number of data features. Empirical tests of our approach show significant improvement in linear classification over both soft-margin SVMs and Expectation Propagation Bayes Point machines for several real-world UCI datasets.

1. Introduction

The Bayesian approach to classification begins with the *Bayes-optimal classifier*. This technique requires evaluating all possible classifiers and allowing them to vote on the classifications of novel inputs, weighted by their performance over the training data and their prior likelihood. More formally, given training data $Z = \{\vec{x}_i, y_i\}$ and novel input vector $\vec{x}$, the Law of Total Probability allows us to express the most likely output class y^* in terms of the marginal over all possible data models:

$$y^* = \operatorname*{argmax}_y P(y|\vec{x}, Z) \tag{1}$$

$$= \operatorname*{argmax}_y \int_{\mathcal{H}} P(y|\vec{x}, h) P(Z|h) P(h) dh \tag{2}$$

Here, $\mathcal{H}$ is the space of all possible models, or hypotheses, of the data. The advantage of the Bayes-optimal classifier is that, on average, it outperforms any other classifier that uses the same hypothesis space $\mathcal{H}$ and prior knowledge (Bishop, 1995). Unfortunately, evaluating all possible classifiers is generally intractable.

Decisions made by the Bayes-optimal classifier may not correspond to any single hypothesis within $\mathcal{H}$. One common way to approximate the Bayes-optimal classifier is to locate a single "average" hypothesis whose behavior resembles the Bayes-optimal classifier as closely as possible, and whose expected generalization error reaches a minima over $\mathcal{H}$. Such classifiers are known as *Bayes Point Machines* (Ruján, 1997).

Consider the case where $\mathcal{H}$ consists of all linear classifiers, so that $h(\vec{x}) = \text{sign}(\vec{w} \cdot \vec{x})$. If needed, the elements of input vectors $\vec{x}$ may be the nonlinear features $\psi : U \to \mathbb{R}$ of some underlying space U, so that $\vec{x} = (\psi_1(u), \cdots, \psi_D(u))$. This allows for nonlinear decision boundaries within the input space U, including affine bias ($\psi_0 \equiv 1$). Data likelihood $P(Z|h)$ can be defined based on the error rate of the classifier h:

$$P(Z|h) \propto \exp(-\hat{C} \sum_{i=1}^{S} L_{01}(y_i\, \vec{w} \cdot \vec{x})) \tag{3}$$

$$L_{01}(x) = I[x < 1] \tag{4}$$

where $I \in \{0, 1\}$ is the indicator function, and S is the number of training samples in Z. $\hat{C}$ is a hyperparameter of the model, which describes the degree of noise present in the training data. The prior $P(h)$ is usually taken to be gaussian: $P(h) = \exp(|\vec{w}|^2/(2\sigma^2))$. The full posterior distribution $P(h|Z)$ can then be given as:

$$P(\vec{w}|Z) \propto \prod_{i=1}^{S} e^{-\beta C I(y_i\, \vec{w} \cdot \vec{x}_i \geq 1)} \prod_{d=1}^{D} e^{-\beta w_d^2/2} \tag{5}$$

with C and β hyperparameters related to $\hat{C}$ and σ.

It has been shown that for linear classifiers, the Bayes-optimal classifier is closely approximated by the output

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

of a the average classifier: $\vec{w}^{opt} = \mathbf{E}_{P(\vec{w}|Z)}[\vec{w}]$ (Herbrich et al., 2001; Ruján, 1997). Unfortunately, $\vec{w}^{opt}$ still requires integrating over $\mathcal{H}$, and remains highly challenging to estimate. Past approaches to finding the mean of the posterior $P(\vec{w}|Z)$ have most commonly used sampling techniques to sample from the *version space* (Ruján, 1997; Herbrich et al., 2001). The version space is the set of linear classifiers that correctly classify every training sample, which is equivalent to $P(\vec{w}|Z)$ as β goes to infinity. This strategy is only available when the training data is separable and free of noise, and so attempts have been made to apply sampling techniques in the non-separable case by approximating $P(\vec{w}|Z)$ as a uniform distribution over an enlarged polyhedra (Herbrich et al., 2001). However, this approximation has not been successful in outperforming the traditional soft-margin SVM.

Note that support vector machines (SVMs) can also be viewed as an approximation of the Bayes point. First, the mean of the posterior is approximated by its mode, the maximum a posteriori (MAP) estimate. Secondly, in place of the 0-1 loss function of equation 4, SVM uses a *surrogate loss function* known as the hinge loss: $L_H(x) = \max(0, 1 - x)$. Because the hinge loss is convex, SVMs can be solved using traditional optimization techniques. This is a powerful advantage for SVMs, because the MAP can be found both quickly and exactly. The disadvantage of the hinge loss is that it is unbounded: the effect of a misclassified point on the decision boundary grows without bound as the point is placed farther away. Thus, classifiers using the hinge loss may be susceptible to outliers. Several attempts have been made to exploit 0-1 loss (Li & Lin, 2007) or a close approximation (Perez-Cruz et al., 2003) for SVMs. Unfortunately, the resulting error function is highly non-convex (Fig. 1a).

Another approach to estimating $\mathbf{E}_{P(\vec{w}|Z)}[\vec{w}]$ is the use of Gaussian expectation propagation (EP) (Minka, 2001; Opper & Winther, 1999). EP is a message passing algorithm for statistical inference; a generalization of assumed density filtering and belief propagation. EP requires that the posterior distribution be approximated by an exponential family distribution. In general, applying EP to the Bayes Point problem requires solving a large integral over the hypothesis space (as in equation 2). When the approximating distribution is Gaussian, a closed form solution to EP messages can be derived via integration by parts. Unfortunately, the Bayes Point posterior $P(\vec{w}|Z)$ is highly non-convex (as in figure 1a), and its coarse structure often exhibits non-negligible skew, so that the Gaussian approximation may not be ideal. While EP methods outperformed hard-margin SVMs, results against soft-margin SVMs were mixed (Opper & Winther, 1999). The Gaussian assumption also inhibits the use of sparse (non-gaussian) priors over $\vec{w}$, including the l_0 norm or the l_1 norm used by LASSO (Tibshirani, 1994). A similar Gaussian approximation is often used by Gaussian process classifiers (Rasmussen & Williams, 2005).

In this paper, we investigate the benefits of estimating the Bayes point without approximating the posterior as a Gaussian. Exactly computing $\vec{w}^{opt}$ is an NP-complete problem, and so we will be forced to use other approximations. In spite of these approximations, in section 5, we show that our approach significantly outperforms both soft-margin SVMs and EP Bayes point machines in six out of seven real-world UCI datasets.

In order to estimate $\vec{w}^{opt}$ without relying on a Gaussian assumption, we use message passing algorithms that pass full vector-valued messages at each iteration. Straightforward application of these message passing techniques for Bayes point estimation would be intractable, and in fact would require *more* computation than a brute force evaluation of the intractable integrand in equation 2. In section 4, we propose a new optimization technique based on linear multiple choice knapsack problems that reduces the computational cost of each message from exponential to linear in the number of features D, while simultaneously reducing numerical error.

2. Formulating SVMs as a Factor Graph

Our goal is to perform probabilistic inference over the posterior of $\vec{w}$ listed in Eq. 5. Unfortunately, this distribution is highly non-convex, and may have many dimensions. Worse yet, it is not smooth; it is non-differentiable at many points. One way to gain leverage against such a problem is to exploit the local structure underlying this probability distribution by expressing it as a *factor graph*. A factor graph is a graphical representation of a probability distribution that has been written as a product of *potential functions* ϕ:

$$p(\vec{X}) = \prod \phi_i(\vec{x_i}) \qquad \vec{x_i} \subseteq \vec{X} \tag{6}$$

Specifically, a factor graph is a bipartite graph in which each potential function $\phi_i(\vec{x_i})$ is represented by a factor node f, which is connected to one variable node v for each element of the vector $\vec{x}_i$.

One way to express equation 5 as a factor graph is to represent each dimension of $\vec{w}$ as a separate variable node. We construct this factor graph to have S factor nodes, one for each training sample. Each factor is

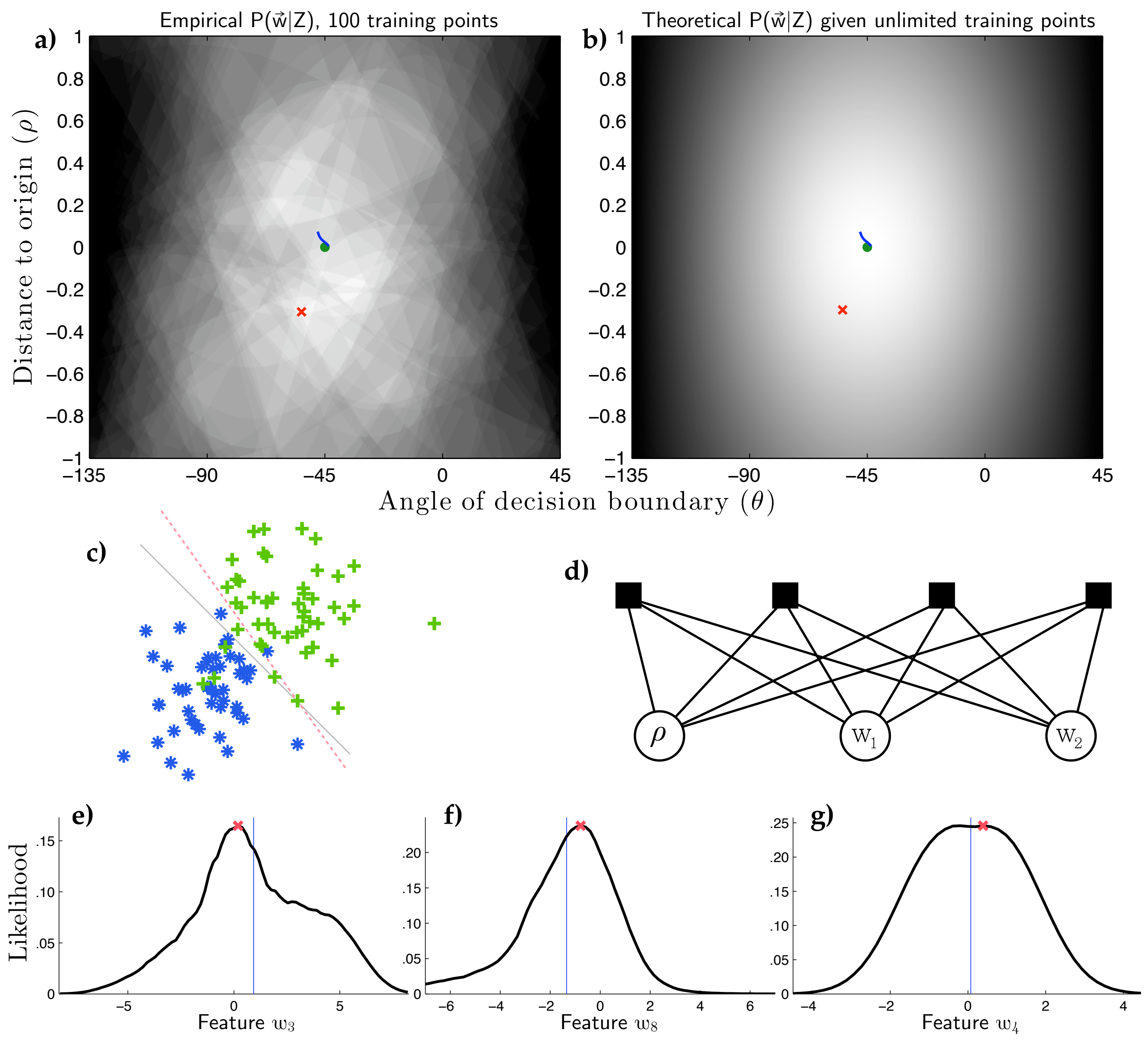

Figure 1. The posterior distribution $P(\vec{w}|Z)$ is often non-smooth and non-Gaussian. Both MAP and Gaussian approximations can lead to errors when estimating the Bayes Point.
a) A plot of the $P(\vec{w}|Z)$ (Eq. 5) for the simple Gaussian training data shown in subfigure c. Each point in this plot corresponds to a different decision boundary (described by its angle θ and offset ρ). The margin width $|\vec{w}|$ is fixed at the width that minimizes error. The green dot represents the ground-truth optimal decision boundary for this data. The red x shows the maximum a posteriori (MAP) estimate over $P(\vec{w}|Z)$. The blue line shows the decision boundary chosen by our knapsack-based algorithm, using values of β ranging from 5 to 50.
b) A plot of the ground-truth test-set performance of each decision boundary, computed using the data's known distribution. Here, theoretical $P(\vec{w}|Z)$ is Gaussian only because Z is Gaussian; non-Gaussian data leads to non-Gaussian $P(\vec{w}|Z)$.
c) Simple toy training data. 50 points each are generated from unit-variance Gaussians centered at $(1, 1)$ and $(-1, -1)$. Ground-truth optimal (solid) and MAP decision boundaries (dashed) are drawn in.
d) A factor graph for inferring the optimal decision hyperplane given four sample points of two-dimensional data. Variable node ρ corresponds to affine bias ($\psi_0 \equiv 1$).
e-g) Three example max-marginals from the Heart and Liver UCI datasets. Often, marginals are sufficiently skewed so that a Gaussian approximation is not accurate, and the MAP estimate (red x) does not lie near the mean (blue line, $\beta = 1$). In other cases, the marginal is flat or even bimodal near its peak, which may also place the MAP far from the mean. These examples illustrate the potential advantages of estimating the mean without making a Gaussian approximation.

connected to each variable node, and has the potential

$$\phi_i(\vec{w}) = \exp(-I[y_i(\vec{w} \cdot \vec{x}_i) \geq 1])^{C\beta} \tag{7}$$

Each variable node w_d also has a single-variate prior with potential function $\phi_d(w_d) = \exp(w_d^2/2)^{\beta}$. An example factor graph is pictured in Figure 1d.

Because this framework learns the weights $\vec{w}$ directly, this approach is constrained to use a linear kernel. A kernelized version can also be constructed, where the variable nodes are the coefficients α_i. The posterior distribution is then given by:

$$P(\vec{\alpha}|Z) = \exp\Bigg(-\frac{1}{2}\sum_{i=1}^{S}\sum_{j=1}^{S}\alpha_i\alpha_j y_i y_j(\vec{x}_i \cdot \vec{x}_j) - C\sum_{i=1}^{S} L_{01}\big(\sum_{j=1}^{S}\alpha_j y_i y_j(\vec{x}_i \cdot \vec{x}_j)\big)\Bigg)^{\beta} \tag{8}$$

While the approach we outline here is applicable to the kernelized version, in this paper we will focus on the non-kernelized formulation of figure 1d. Note that using random features often allows low-dimensional linear classifiers to outperform large-scale kernel machines (Rahimi & Recht). While kernelizing our approach is feasible, it may not always be necessary.

3. Convergent Message Passing

Highly factorized distributions such as Eq. 5 are often optimized using message passing techniques, such as belief propagation, expectation propagation,or TRW-S. These methods exploit the factorization of the distribution to achieve efficient inference. As mentioned earlier, Gaussian expectation propagation, which approximates the posterior as a Gaussian, has been applied to this problem before (Minka, 2001).

In this paper, we explore the application of message passing algorithms that do not rely on a Gaussian approximation of the posterior. The primary reason that such methods have never been applied to the Bayes point problem is the highly connected nature of the underlying factor graph. A straightforward application of traditional message passing algorithms would not provide any advantage in computational complexity over a brute-force solution (i.e. the evaluation of all possible classifiers). In section 4, we introduce a new computational shortcut that reduces the complexity of computing such messages from exponential to linear in the number of features D.

The message passing algorithm we apply in this paper is a convergent variant of belief propagation due to Heskes (Heskes et al., 2003). Like belief propagation, Heskes' algorithm is an algorithm that utilizes the factorization of a distribution (Eq. 6) to perform probabilistic inference. In its *sum-product* form, Heskes' algorithm computes estimates the single-variate marginals $\mu_i(x_i) = \sum_{X\setminus x_i} p(\vec{x})$ by iteratively computing vector-valued messages along each edge of the factor graph. The mean of the distribution can be computed easily from these marginals. In its *max-product* form, Heskes' algorithm estimates the single-variate *max-marginals* $\nu_i(x_i) \propto \max_{X\setminus x_i} p(\vec{x})$.

Heskes' algorithm was developed in response to shortcomings of the original belief propagation algorithm. The original formulation of belief propagation was limited in that it was not guaranteed to converge in graphs containing loops. The theoretical justifications of belief propagation were later greatly advanced when it was shown that fixed points of belief propagation were the minima of the Bethe free energy, a quantity from statistical physics which can be thought of as an approximate measure of the distance between a multivariate probability distribution and a set of marginals (Yedidia et al., 2000). Furthermore, this discovery has lead to a variety of new methods that minimize Bethe free energy directly, and are guaranteed to converge (Yuille, 2002), including Heskes' algorithm. These variants typically benefit not only from guaranteed convergence, but also improved performance.

Like all belief propagation algorithms, Heskes' algorithm proceeds by iteratively passing vector-valued messages m along each edge in the factor graph. These include messages from variable nodes to factor nodes $m^t_{i\to f}(x_i)$, and messages from factor nodes to variable nodes $m^t_{f\to i}(x_i)$, as given below:

$$m^t_{i\to f}(x_i) = m^t_{i\to f}(x_i)^{\frac{1-n_i}{n_i}} \prod_{g\in\mathcal{N}(i)\setminus f} m^{t-1}_{g\to i}(x_i)^{\frac{1}{n_i}} \tag{9}$$

$$m^t_{f\to i}(x_i) = \int_{\vec{x}_{\mathcal{N}(f)\setminus i}} \hat{\phi}_f\left(\vec{x}_{\mathcal{N}(f)}\right) \prod_{j\in\mathcal{N}(f)\setminus i} m^t_{j\to f}(x_j)\, d\vec{x} \tag{10}$$

$$b^t_i(x_i) \propto \prod_{g\in\mathcal{N}(i)} m^t_{g\to i}(x_i) \tag{11}$$

where f and g are factor nodes, i and j are variable nodes, $\mathcal{N}(i)$ is the set of neighbors of node i, and $n_i = |\mathcal{N}(i)|$. Here, $b_i(x_i)$ is the estimated marginal of variable i, also known as the *beliefs* of node i. In Heskes' variant of belief propagation, $\tilde{\phi}_f$ is initialized to ϕ_f. Equations 9 and 11 are then iterated until convergence is reached. At this time, $\tilde{\phi}_f$ is updated by multiplying with $b^{\tau}_j(x_j)^{(n_j-1)/n_j}$, where b^{τ}_j is the beliefs of variable node j at the time of this convergence. Equations 9 and 11 are then iterated to convergence again, and the process is repeated until the potentials $\tilde{\phi}_f$ converge.

Convergent max-product belief propagation is achieved by replacing the integrand of equation 10 with a maximization:

$$m^t_{f\to i}(x_i) = \max_{\vec{x}_{\mathcal{N}(f)\setminus i}} (\tilde{\phi}_f(\vec{x}_{\mathcal{N}(f)}) \prod_{j\in\mathcal{N}(f)\setminus i} m^t_{j\to f}(x_j)) \quad (12)$$

Notice that the beliefs and messages require us to represent probability distributions over the real-valued weights w_d. Most often, this is achieved by discretizing the possible values of each w_d, and representing beliefs and messages using histograms. In the experiments in this paper, we use adaptive histograms where the discretization is adjusted after each iteration to best capture the distribution, as done in (Potetz, 2007).

4. Efficient Message Passing for 0-1 Potential Functions Using L-MCKP

The computational bottleneck of belief propagation is the integral in equation 10; computing messages $m^t_{f\to i}$ takes $\mathcal{O}(M^{n_f})$ operations, where M is the number of possible states of each variable node, and n_f is the number of variable nodes neighboring on f. For the bipartite factor graph in figure 1d, each message $m^t_{f\to i}$ requires $\mathcal{O}(M^D)$ operations. At each iteration, a message must be passed along every edge, making each iteration of Heskes' algorithm $\mathcal{O}(SDM^D)$ for figure 1d. By contrast, suppose we were performing an exhaustive search for a linear classifier $\vec{w}$ given S samples of D-dimensional data, and we discretized the search space so that each variable w_d can take on one of M possible values. An exhaustive search of this space would require $\mathcal{O}(SM^D)$ operations, and computing the single-variate marginals or max-marginals would in general take $\mathcal{O}(SDM^D)$ operations. Thus, for estimating the Bayes point, a naive implementation of belief propagation does not provide any computational advantages over the brute force solution.

In this section, we describe a technique to reduce the complexity of computing each message from $\mathcal{O}(M^D)$ to $\mathcal{O}(MD)$, while at the same time increasing accuracy and numerical stability. Our method works for the max-product form of Heskes' algorithm, which means we are approximating the Bayes point as the mean of the max-marginal:

$$w_i^{opt} = \sum_{\vec{w}} w_i P(\vec{w}|Z) \precapprox \sum_{w_i} w_i (\max_{\vec{w}\setminus w_i} P(\vec{w}|Z)) \quad (13)$$

Thus, the result of our algorithm is somewhere between the MAP computed by SVMs and the true Bayes Point. Like the Bayes point, our estimate approaches the MAP as β goes to infinity. The argument in favor of making this approximation is that, when inferring w_i^{opt}, skew over the marginal $P(w_i|Z)$ is much more likely to distort estimates of w_i than skew over $P(w_j|Z)$. Thus, for each element i of $\vec{w}^{opt}$, we treat the final marginalization as the most important, and approximate only over the earlier marginalizations. In section 5, we show promising results in comparison with both soft-margin SVMs and EP Bayes point machines, which suggests that this approximation may have some advantages over the Gaussian approximation made by EP or the MAP used by SVM.

Our approximation has a significant additional benefit: because exponentiation commutes across maximization, $\max P(\vec{w}|Z) = (\max P(\vec{w}|Z)^{1/\beta})^\beta$. Thus, hyperparameter β can be estimated by n-fold cross-validation *without recomputing the max-marginals.* In contrast, a sum-product approach would require reperforming message passing for each value of β.

For max-product message passing, the computational bottleneck is the multidimensional maximization in equation 12. Let us write $m_i(w_i)$ and $M_i(w_i)$ as shorthand for $\log m^t_{f\to i}(w_i)$ and $\log m^t_{i\to f}(w_j)$ respectively. Substituting in our definition of ϕ (equation 7), we get:

$$m_i(w_i^*) = \max(\max_{\vec{w}\in\mathcal{A}} \sum_{j\in\mathcal{N}(f)\setminus i} M_j(w_j), \quad (14)$$

$$-\beta C + \max_{\vec{w}\in\mathcal{B}} \sum_{j\in\mathcal{N}(f)\setminus i} M_j(w_j)) \quad (15)$$

$$\mathcal{A} = \{\vec{w} \,|\, w_i = w_i^*,\, \vec{w}\cdot\vec{x}_f \geq 1\} \quad (16)$$

$$\mathcal{B} = \{\vec{w} \,|\, w_i = w_i^*\} \quad (17)$$

where the vector $\vec{x}_f$ is the training point associated with factor node f. Intuitively, $\mathcal{A}$ is the set of weight vectors $\vec{w}$ that classify $\vec{x}_f$ correctly, and whose i^{th} component is given by w_i^*. The maximization under condition $\mathcal{B}$ can be found trivially in $\mathcal{O}(DM)$ time by maximizing each $M_j(w_j)$ independently. The maximization under condition $\mathcal{A}$ can be seen to be equivalent to a *multiple-choice knapsack problem* (MCKP). In this variant of the classic knapsack problem, we are given D lists of M items, each with a weight $\mathbf{w}_{di}$ and a profit $\mathbf{p}_{di}$. We must fill a knapsack by choosing one item from each list so as to maximize total profit without exceeding the capacity of the knapsack c:

$$\text{maximize profit:} \quad \sum_{d=1}^{D}\sum_{k=1}^{M} \alpha_{dk}\mathbf{p}_{dk} \quad (18)$$

$$\text{subject to:} \quad \sum_{d=1}^{D}\sum_{k=1}^{M} \alpha_{dk}\mathbf{w}_{dk} \leq c \quad (19)$$

$$\alpha_{dk} \in \{0,1\} \quad \text{and} \quad \forall d,\ \sum_{k=1}^{M} \alpha_{dk} = 1 \quad (20)$$

The weights and prices are set according to $\mathbf{p}_{dk} = M_d(w_{dk}) + \bar{\mathbf{p}}_d$ and $\mathbf{w}_{dk} = -x_d^f w_{dk} + \bar{\mathbf{w}}_d$, where w_{dk} is the center of the k^{th} histogram bin of variable w_d, and x_d^f is the d^{th} feature of training point x_f. $\bar{\mathbf{p}}_d$ and $\bar{\mathbf{w}}_d$ are constants to ensure that weights and prices are never negative. Total capacity c is given by $c = x_i^f w_i^* - 1 + \sum_{d=1}^{D} \bar{\mathbf{w}}_d$.

The multiple choice knapsack problem is often solved using dynamic programming. One such method resembles a special case of the *linear constraint node* technique for efficient belief propagation (Potetz, 2007). The linear constraint node method would allow us to compute each message in time $\mathcal{O}(SD^2M^2)$. Unfortunately, given the large number of messages that must be computed, a procedure that is quadratic in the number of histogram bins M remains prohibitively slow. Despite our adaptive procedures for placing the histogram bins of each message, we have found that M must be set relatively high to accurately represent messages and achieve good performance. In the experiments of section 5, M is fixed at 128. In the remainder of this section, we demonstrate a method for computing messages that is linear with respect to M, thus improving speed by over 50-fold. At the same time, our method uses implicit interpolation to reduce the level of discretization error that is caused by representing messages $m_i(w_i)$ and $M_i(w_i)$ as histograms.

One tool that is commonly used in solving MCKPs is to compute an upper bound on the maximum profit by allowing knapsack items to be divided into pieces. Specifically, we relax the constraint in equation 20. This is known as the *linear multiple choice knapsack problem* (L-MCKP). It can be shown that an optimal solution to L-MCKP can always be found that subdivides at most one item (i.e. at most one α_{di} is fractional) (Kellerer et al., 2004).

In the special case that each message $M_j(w_j)$ is concave (i.e. $\frac{\mathbf{p}_{d2}-\mathbf{p}_{d1}}{\mathbf{w}_{d2}-\mathbf{w}_{d1}} \geq \frac{\mathbf{p}_{d1}-\mathbf{p}_{d0}}{\mathbf{w}_{d1}-\mathbf{w}_{d0}}$), solutions to the L-MCKP give the global maximum of:

$$\max_{\vec{w}\in\mathcal{A}} \sum_{j\in\mathcal{N}(f)\setminus i} \hat{M}_j(w_j) \tag{21}$$

where $\hat{M}_j(w_j)$ is a continuously-valued function given by the linear interpolation of $M_j(w_j)$ (Kellerer et al., 2004). For our purposes, this approximation is not only acceptable, but desirable; while the joint distribution $P(\vec{w}|Z)$ is merely piece-wise continuous (as in figure 1a), single-variate max-marginals over that distribution tend to be quite smooth. Thus, interpolating the messages $M_j(w_j)$ and solving in the continuous domain is an improvement over solving the discrete problem.

In general, L-MCKP can be solved in $\mathcal{O}(DM\log(DM))$ time using a greedy algorithm (Kellerer et al., 2004). This algorithm first sorts each list by weight, and then constructs D new lists of $M-1$ items, with weights $\mathbf{w}'_{d,i} = \mathbf{w}_{d,i} - \mathbf{w}_{d,i-1}$ and profits $\mathbf{p}'_{d,i} = \mathbf{p}_{d,i} - \mathbf{p}_{d,i-1}$. These items are then merged into a single list, and sorted by item *efficiency* $\mathbf{p}'_{d,i}/\mathbf{w}'_{d,i}$. We can then scan this list, iterating until the total weight $\sum_{k=1}^{k'} \mathbf{w}'_{d,i}$ reaches capacity c. The sum profit up to this point (plus the fractional profit of the final item) is equal to the L-MCKP solution.

The computational bottleneck of this algorithm is sorting the item lists into a single list, which typically requires $\mathcal{O}(DM\log(DM))$ operations. Histograms are always stored sorted by item weight $\mathbf{w}$ so for concave messages, each of the D lists is already sorted. Thus, we only need to merge the sorted lists together, which reduces the algorithm to $\mathcal{O}(DM\log D)$. We also need to recompute the L-MCKP solution for each value of w_i^*. Fortunately, the sorted item list can be reused for each computation. Thus, when all messages are concave, we can derive $m_i(w_i^*)$ in equation 15 in $\mathcal{O}(DM\log D)$ time.

When not all messages are concave, we can still take advantage of the above computational shortcuts by using a dynamic programming framework. One dynamic programming approach to solving the MCKP requires recursively computing the maximum profit of choosing one item from the first K lists:

$$T_K(c_K) = \max_{\vec{\alpha}} \sum_{d=1}^{K}\sum_{k=1}^{M} \alpha_{dk}\mathbf{p}_{dk} \tag{22}$$

$$\text{subject to:} \quad \sum_{d=1}^{K}\sum_{k=1}^{M} \alpha_{dk}\mathbf{w}_{dk} \leq c_K \tag{23}$$

Solving for each T_K is equivalent to solving the MCKP on two item lists: M_K and T_{K-1}. If we order the messages so the first K' messages are concave, then $T_{K'}$ can be solved efficiently (and without first computing $T_{K'-1}$) using the techniques above. Finally, we can utilize the L-MCKP shortcut to more quickly compute each T_K for nonconcave messages as well. Recall that computing T_K requires solving a MCKP of two messages. It can be shown that there is a solution $(\hat{w}_1, \hat{w}_2)$ to

$$(\hat{w}_1, \hat{w}_2) = \operatorname*{argmax}_{w_1 x_1^f + w_2 x_2^f < c_1} \left(\hat{M}_1(w_1) + \hat{M}_2(w_2)\right) \tag{24}$$

such that both $\hat{w}_1$ and $\hat{w}_2$ lie within *undominated* regions of $\hat{M}_1$ and $\hat{M}_2$ (Kellerer et al., 2004). A point t of $\hat{M}_1$ is dominated if $\hat{M}_1(t) \leq \hat{M}_1(s)$ for some $s < t$. It can also be shown that $\hat{w}_1$ and $\hat{w}_2$ must satisfy one of two conditions. The first possibility is that $\hat{M}_1$ and

Table 1. Error rates for linear classification over test sets ± the standard deviation of the mean over 500 test/training splits for soft-margin (hinge loss) linear SVMs, Expectation Propagation (source code from (Minka, 2001)), and our knapsack programming approach (Knap). Entries in bold show statistically significant improvement over the other two methods ($p < 0.01$). The SIZE row lists the number of samples S and the number of features D, respectively.

	LIVER	SONAR	HEART	CONTRACEPT.	PIMA	AUSTRALIAN	BREAST
SIZE	345; 7	208; 61	270; 14	1473; 10	768; 9	690; 15	683; 11
SVMS	31.93 ± 0.23	24.18 ± 0.27	16.05 ± 0.21	32.10 ± 0.11	23.02 ± 0.14	13.56 ± 0.12	3.26 ± 0.06
EP	32.60 ± 0.24	23.21 ± 0.29	16.25 ± 0.21	32.49 ± 0.11	23.15 ± 0.14	13.32 ± 0.12	3.24 ± 0.06
KNAP	$\mathbf{30.52 \pm 0.22}$	$\mathbf{21.74 \pm 0.29}$	$\mathbf{15.60 \pm 0.21}$	$\mathbf{31.66 \pm 0.11}$	23.74 ± 0.14	$\mathbf{13.00 \pm 0.12}$	$\mathbf{3.11 \pm 0.06}$
$\beta = \infty$	32.88 ± 0.25	24.39 ± 0.28	15.84 ± 0.21	32.08 ± 0.11	23.93 ± 0.15	15.06 ± 0.13	4.35 ± 0.08

$\hat{M}_2$ are both concave at $\hat{w}_1$ and $\hat{w}_2$, respectively. In this case, the maxima is equal to the L-MCKP solution for two item lists consisting of the concave regions of $\hat{M}_1$ and $\hat{M}_2$. If R is the number of concave regions in each of the two messages, then we can compute the L-MCKP solution for all pairs of concave regions of $\hat{M}_1$ and $\hat{M}_2$ in $\mathcal{O}(R^2(M/R)) = \mathcal{O}(RM)$ time. In practice, we have found that messages $M_j(w_j)$ are concave most of the time, and those that are not concave typically have a value of R of at most two or three.

The second possibility for $(\hat{w}_1, \hat{w}_2)$ is that (without loss of generality) $\hat{M}_1$ is nonconcave at $\hat{w}_1$, while $\hat{w}_2$ occurs at the edge of an undominated region of $\hat{M}_2$. Since there are $2R$ such extrema, we can check all such possibilities in $\mathcal{O}(RM)$ time. Thus, we can find the solution to equation 24 in $\mathcal{O}(RM)$ time. Since fewer than D intermediate functions T_K must be computed, we can then compute the output message $m^t_{f\to i}(w_i)$ in $\mathcal{O}(RDM)$ time using dynamic programming and L-MCKP together. This brings the total computational complexity of Heskes' algorithm to $\mathcal{O}(RSD^2M)$ per iteration for the factor graph in figure 1d. Note that the kernelized factor graph of equation 8 is also eligible for the computational shortcuts introduced here, bringing the runtime of the kernelized approach to $\mathcal{O}(RS^3M)$ per iteration. By comparison, the complexity of traditional dual-decomposition SVM algorithms is $\Omega(S^2D)$ (Bottou & Lin, 2007; Shalev-Shwartz & Srebro, 2008).

5. Results

Table 1 shows the results of our approach on seven datasets taken from the UCI machine learning repository (Asuncion & Newman, 2007). Our knapsack algorithm obtained a substantial improvement over both soft-margin SVMs and EP in six of those seven datasets (in each case, $p < 0.001$ under both Wilcoxon and paired Student's t-tests). Each dataset was randomly split 500 times into a training set (80% of samples) and a test set (20% of samples). Each training set was preprocessed to have unit variance in each dimension and low bias ρ. For each algorithm, all hyperparameters (C and/or β) were found by performing 5-fold cross-validation independently on each of the 500 trials. Thus, for each trial, hyperparameter selection is performed with no foreknowledge of the test set. An alternative (and common) evaluation procedure is to choose a single value of the hyperparameters to use for all 500 trials; under this procedure the performance advantage of the knapsack method increases.

The results in table 1 were achieved after 50 iterations of message passing. Empirically we found that results did not improve significantly after that. As an example of running-time, training a classifier for the *Liver* dataset took a total of 12 seconds on a 2.8GHz Intel Xeon processor, and 57 seconds for the larger *Contraceptives* dataset. Even though belief propagation is a highly parallelizable algorithm, our implementation is single-threaded, suggesting that significant speed boosts are still possible.

Because the our algorithm uses the 0-1 loss function, we expected high robustness towards outliers. To test this, we added a single outlier at position $(10, 0)$ to the data from figure 1c. While SVM error rate increased from 7.87% to 7.94% after adding the outlier, the knapsack algorithm remained steady at 7.87%.

The final row of table 1 shows the results of our algorithm when β (Eq. 5 and 13) is fixed at infinity instead of chosen via cross-validation. When β goes to infinity, our approach approximates the MAP under the 0-1 loss function. The difference in performance shows that not all of the improvement of our algorithm was due to the 0-1 loss function, and it demonstrates the benefit of computing the means of max-marginals. In contrast, if β is fixed at one, performance gains remain statistically significant over the same six datasets.

6. Discussions and Conclusions

The methods introduced in this paper are well-suited for a number of extensions to the binary classifier model. First, because Heskes' algorithm is a discrete optimization technique, it can be used to handle discrete or categorical data very naturally. Secondly, it is popular to consider non-Gaussian priors over weights $\vec{w}$ as a method of feature selection. For example, the LASSO method minimizes $\vec{w}$ over the l_1 norm (Tibshirani, 1994). This is equivalent to using Laplace distributions as priors over $\vec{w}$. Because belief propagation does not rely on the convexity of the posterior, the methods discussed in this paper are equally compatible with any prior model: minimizing $\vec{w}$ over the l_0 or l_1 norm is no more difficult that minimizing over the l_2 norm. Thirdly, other machine learning tasks such as transductive Gaussian process classifiers, boosting, or training neural networks have similar optimization criteria that are eligible for the approach introduced here. Finally, Bayesian approaches to classification have several additional advantages that were not explored in this paper. In principle, the hyperparameters C and β can be inferred via probabilistic inference rather than optimized using cross-validation (Hernández-Lobato & Hernández-Lobato, 2008). Confidence intervals can be computed for classifications. These issues are all subjects for future research.

We hope that the results of this paper will motivate further efforts to efficiently estimate the Bayes Point without Gaussian approximation. We expect that such approaches will be most beneficial under conditions where the posterior distribution $P(\vec{w}|Z)$ is highly non-Gaussian or non-convex. These circumstances are likely to arise when the data itself is highly non-Gaussian (resulting in a posterior with a non-Gaussian shape), or when there are few available training samples (resulting in a highly discontinuous posterior). We expect that advances can be made in predicting, from the data, the likely amount of benefit to pursuing a more accurate estimate of the Bayes point classifier.

References

Asuncion, A. and Newman, D. J. UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Nov 1995.

Bottou, L. and Lin, C.J. Support vector machine solvers. In Bottou, L., Chapelle, O., DeCoste, D., and Weston, J. (eds.), *Large Scale Kernel Machines*, pp. 301–320. MIT Press, Cambridge, MA., 2007.

Herbrich, R., Graepel, T., and Campbell, C. Bayes point machines. *J. Mach. Learn. Res.*, 1:245–279, 2001.

Hernández-Lobato, D. and Hernández-Lobato, J. M. Bayes machines for binary classification. *Pattern Recogn. Lett.*, 29:1466–1473, July 2008.

Heskes, T., Albers, K., and Kappen, B. Approximate inference and constrained optimization. In *UAI*, pp. 313–320, 2003.

Kellerer, H., Pferschy, U., and Pisinger, D. *Knapsack Problems*. Springer, 2004.

Li, L. and Lin, H.T. Optimizing 0/1 loss for perceptrons by random coordinate descent. In *Proceedings of the 2007 International Joint Conference on Neural Networks*, pp. 749–754, 2007.

Minka, T. P. Expectation propagation for approximate bayesian inference. In *UAI 2001*, pp. 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Opper, M. and Winther, O. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12:2000, 1999.

Perez-Cruz, F., Navia-Vazquez, A., Figueiras-Vidal, A. R., and Artes-Rodriguez, A. Empirical risk minimization for support vector classifiers. *Neural Networks, IEEE Transactions on*, 14(2):296–303, 2003.

Potetz, B. Efficient belief propagation for vision using linear constraint nodes. In *CVPR 2007*. IEEE Computer Society, Minneapolis, MN, USA, 2007.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *NIPS 2007*.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, December 2005.

Ruján, P. Playing billiards in version space. *Neural Comput.*, 9(1):99–122, 1997.

Shalev-Shwartz, S. and Srebro, N. SVM optimization: inverse dependence on training set size. In *ICML 2008*, pp. 928–935, New York, NY, USA, 2008. ACM.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. Generalized belief propagation. In *NIPS*, pp. 689–695, 2000.

Yuille, A. L. CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. *Neural Computation*, 14(7): 1691–1722, 2002.